Информатика и вычислительная техника

УДК 519.863 **DOI: 10.14529/ctcr160201**

ЗАДАЧА ОПТИМИЗАЦИИ ВЫЧИСЛЕНИЙ В СЕТЕВЫХ СТРУКТУРАХ

В.Н. Бурков[1], Е.В. Ляпунцова[2, 3], Р.С. Шихалиев[4]

[1] Институт проблем управления им. В.А. Трапезникова РАН, г. Москва,
[2] Комитет Совета Федерации по социальной политике, г. Москва,
[3] МРО «Лига преподавателей высшей школы», г. Москва,
[4] Московский государственный университет путей сообщения (МИИТ), г. Москва

Рассматривается вычислительная сеть из n вершин (это вершины, в которых решаются те или иные задачи), m входных вершин и m выходных вершин (m – число решаемых задач). Каждой задаче соответствует путь в сети с входом H и выходом K, соответствующий некоторому алгоритму решения задачи. Одновременно в каждом узле может решаться только одна задача. Поэтому может возникнуть конфликтная ситуация, когда в момент прихода в вершину некоторой задачи эта вершина занята решением другой задачи. Узлы, в которых могут решаться несколько задач, будем называть проблемными. Рассматриваются задачи составления расписания выполнения задач по критериям минимизации времени решения всех задач и минимизации максимального отклонения от требуемых времен решения. Для их решения предложены методы локальной оптимизации, ветвлений, ветвей и границ.

Ключевые слова: сетевые структуры, проблемные узлы, задача составления расписаний, локальная оптимизация.

Введение

Задачи оптимизации вычислений в сетевых структурах различного вида, включая параллельные вычисления, стали весьма популярными в последнее время [1]. В статье рассматривается вычислительная сеть, в которой решается некоторое множество задач. Алгоритму решения каждой задачи соответствует путь в сети. Одновременно в каждой вершине сети может решаться только одна задача, поэтому возможны конфликтные ситуации, когда задача приходит в очередную вершину, требуемую по алгоритму, а та уже занята решением другой задачи.

В статье рассматриваются постановки задач построения расписаний выполнения задач, оптимальных по различным критериям. Предложены алгоритмы решения, в основе которых лежат методы локальной оптимизации, ветвлений и ветвей и границ.

1. Постановка задачи

Рассмотрим вычислительную сеть из n «активных» вершин (вершины, в которых выполняются задачи), m входных вершин и m выходных вершин (по числу решаемых задач). Каждой задаче i соответствует путь частичной подсети, соединяющий вход H_i с выходом K_i, определяющий некоторый способ (алгоритм) решения i-й задачи. И наоборот, способу решения i-й задачи соответствует путь, соединяющий вход с выходом. Пример вычислительной сети приведен на рис. 1 (в скобках у дуг указаны номера решаемых задач).

Обозначим τ_{ij} – время решения подзадачи j в вершине i (указаны в нижних половинах вершин на рис. 2). Одновременно в каждой вершине может решаться только одна задача, поэтому может возникнуть конфликтная ситуация, когда в момент прихода в вершину некоторой задачи вершина занята решением другой задачи. Вершины, в которых решаются несколько задач, назовем проблемными.

Для разрешения конфликтов определим приоритеты подзадач в каждой проблемной вершине (i, j). Эти приоритеты задаются перестановкой $\pi_{ij} = (j_1, j_2, \ldots, j_s)$, где s – число подзадач, которые

решаются в вершине (i, j). Подзадача i_q имеет приоритет перед задачей i_p, если $q < p$. Задание приоритетов позволяет определить моменты T_j, $j = \overline{1,m}$ завершения решения всех задач.

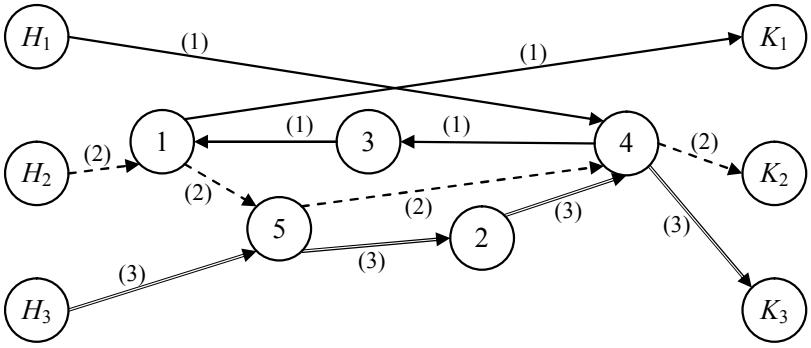

Рис. 1

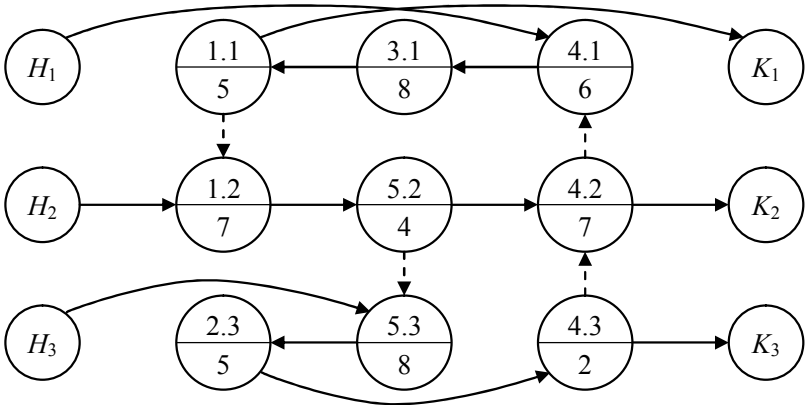

Рис. 2

Предварительно преобразуем сеть рис. 1 к виду, удобному для дальнейших вычислений. Разделим каждую проблемную вершину на несколько вершин – по числу решаемых в ней задач. Эти вершины соединим пунктирными ребрами, которые будем называть ресурсными зависимостями. Задание приоритетов соответствует заданию ориентации ресурсных зависимостей. Преобразованная сеть представлена на рис. 2. Первое число в верхней половине – это номер вершины, второе – номер задачи, число в нижней половине – время решения задачи.

Опишем алгоритм определения моментов завершения выполнения задач при заданных приоритетах π_{ij}.

Описание алгоритма.

1 шаг. Определяем моменты завершения λ_{ij} пути для всех вершин, не учитывая ресурсных зависимостей. Это просто сумма продолжительностей τ_{ij} от начальной вершины до рассматриваемой.

2 шаг. Для каждой проблемной вершины корректируем моменты завершения соответствующих подзадач, учитывая ресурсные зависимости. Опишем алгоритм корректировки (номер вершины опускаем). Пусть в проблемной вершине решаются s подзадач и приоритеты $\pi = (1, 2, \ldots, s)$.

Момент завершения подзадачи 1 равен $\lambda'_1 = \lambda_1$.

Пусть определены моменты завершения решения подзадач 1, 2, …, p. Упорядочим их по возрастанию, т. е.

$\lambda'_{j_1} < \lambda'_{j_2} < \cdots < \lambda'_{j_p}$.

Определяем минимальный номер k такой, что

$\max\left(\lambda_{j_{k-1}}; \lambda_{p+1} - \tau_{p+1}\right) + \tau_{p+1} \le \lambda'_{j_k} - \tau_{j_k}$.

Момент завершения задачи $(p+1)$

$\lambda'_{p+1} = \max\left(\lambda_{j_{k-1}}; \lambda_{p+1} - \tau_{p+1}\right) + \tau_{p+1}$.

Далее повторяем шаги 1 и 2 до тех пор, пока индексы не установятся.

Пример 1.

1 шаг. Имеем:

Задача 1: $\lambda_{41} = 6$, $\lambda_{31} = 14$, $\lambda_{11} = 19$;

Задача 2: $\lambda_{12} = 7$, $\lambda_{52} = 11$, $\lambda_{42} = 18$;

Задача 3: $\lambda_{53} = 8$, $\lambda_{23} = 13$, $\lambda_{43} = 15$.

2 шаг. Приоритеты задач указаны пунктирными дугами на рис. 2. Рассматриваем проблемную вершину 1. Имеем

$\lambda^1_{11} = \lambda_{11} = 19$,

$\lambda^1_{12} = \lambda_{12} = 7$,

так как $\tau_{12} = 7 < \lambda^1_{11} - \tau_{11} = 14$.

Рассматриваем проблемную вершину 4. Имеем

$\lambda^1_{43} = \lambda_{43} = 15$,

$\lambda^1_{42} = \lambda^1_{43} + \tau_{42} = 22$,

$\lambda^1_{41} = \lambda_{41} = 6$,

так как $\lambda^1_{42} - \tau_{42} = 13 > \tau_{41}$.

Рассматриваем проблемную вершину 5. Имеем

$\lambda^1_{52} = \lambda_{52} = 11$,

$\lambda^1_{53} = \lambda^1_{52} + \tau_{53} = 19$,

так как $\lambda^1_{52} - \tau_{52} = 7 > \tau_{53}$.

Повторяем шаги 1 и 2.

1 шаг. Имеем:

Задача 1: $\lambda^2_{41} = \max\left(\lambda^1_{41}; \lambda_{41}\right) = 6$,

$\lambda^2_{31} = \lambda^2_{41} + \tau_{31} = 14$,

$\lambda^2_{11} = \lambda^2_{31} + \tau_{11} = 19$;

Задача 2: $\lambda^2_{12} = 7, \lambda^2_{52} = 11, \lambda^2_{42} = 22$;

Задача 3: $\lambda^2_{53} = 19, \lambda^2_{23} = 24, \lambda^2_{43} = 26$.

Так как $\lambda^2_{42} = 22 > \lambda^0_{42} = 18; \lambda^2_{43} = 26 > 15$, то повторяем шаг 2.

2 шаг. Имеем

Задача 1: $\lambda^3_{41} = 6, \lambda^3_{31} = 14, \lambda^3_{11} = 19$;

Задача 2: $\lambda^3_{12} = 7, \lambda^3_{52} = 11, \lambda^3_{42} = 18$;

Задача 3: $\lambda^3_{53} = 19, \lambda^3_{23} = 24, \lambda^3_{43} = 26$.

Повторение шагов 1 и 2 дает те же значения индексов. Алгоритм закончен.

Рассмотрим две постановки задач.

Задача 1. Определить приоритеты задач всех проблемных вершин, при которых время решения всех задач

$$T = \max_i T_i \to \min. \quad (1)$$

Задача 2. Определить приоритеты задач всех проблемных вершин, при которых максимальное отклонение времен решения задач T_i то требуемых Q_i

$$\Delta = \max_i \left(T_i - Q_i\right) \to \min. \quad (2)$$

Рассмотрим методы решения поставленных задач. Все они относятся к сложным задачам дискретной оптимизации, не имеющим эффективных точных методов решения.

2. Метод локальной оптимизации

Рассмотрим сначала задачу 1. Для применения метода локальной оптимизации в первую очередь необходимо определить понятие окрестности решения. Для этого рассмотрим некоторое решение, задаваемое приоритетами π. Обозначим L_i – путь минимальной длины для i-й задачи, M – множество задач, для которых достигается $T = \max_i T_i$. Очевидно, что для улучшения решения следует изменить приоритеты задач в множестве M.

Согласно методу локальной оптимизации для каждого решения окрестности определяем значение $T(\pi_i)$ критерия (1). Если

$$T(\pi_q) = \min_i T(\pi_i) < T(\pi_0),$$

то переходим к решению π_q и повторяем шаг алгоритма. Если $T(\pi_q) = T(\pi_0)$, то получено локально-оптимальное решение. Далее можно выбрать другое начальное решение и повторить шаг алгоритма.

Алгоритм формирования окрестности для последующей локальной оптимизации рассмотрим на примере.

Пример 2. Рассмотрим сеть рис. 2 с приоритетами задач, указанных во всех проблемных вершинах. Множество M содержит одну задачу 3, причем $T_{max} = 26$. Задача 3 связана ресурсными зависимостями с задачей 2 (проблемные вершины 4 и 5). Рассмотрим вариант изменения приоритетов задач. Возможны 3 варианта:

1. Изменяем приоритеты задач в вершине 5, т. е. $\pi_5 = (3, 2)$.
2. Изменяем приоритеты задач в вершине 4, т. е. $\pi_4 = (2, 3)$.
3. Изменяем приоритеты задач в вершинах 5 и 4, т.е. $\pi_5 = (3, 2)$, $\pi_4 = (2, 3)$.

Вариант 1. Соответствующая сеть приведена на рис. 3.

Установившиеся индексы равны следующим величинам:

Задача 1: $\lambda_{41} = 6$, $\lambda_{31} = 14$, $\lambda_{11} = 19$;

Задача 2: $\lambda_{12} = 7$, $\lambda_{52} = 12$, $\lambda_{42} = 22$;

Задача 3: $\lambda_{53} = 8$, $\lambda_{23} = 13$, $\lambda_{43} = 15$.

Имеем: $T_{max} = 22 < 26$.

Вариант 2. Соответствующая сеть приведена на рис. 4.

Установившиеся индексы приведены ниже:

Задача 1: $\lambda_{41} = 6$, $\lambda_{31} = 14$, $\lambda_{11} = 19$;

Задача 2: $\lambda_{12} = 7$, $\lambda_{52} = 11$, $\lambda_{42} = 18$;

Задача 3: $\lambda_{53} = 19$, $\lambda_{23} = 24$, $\lambda_{43} = 26$.

Имеем: $T_{max} = 26$.

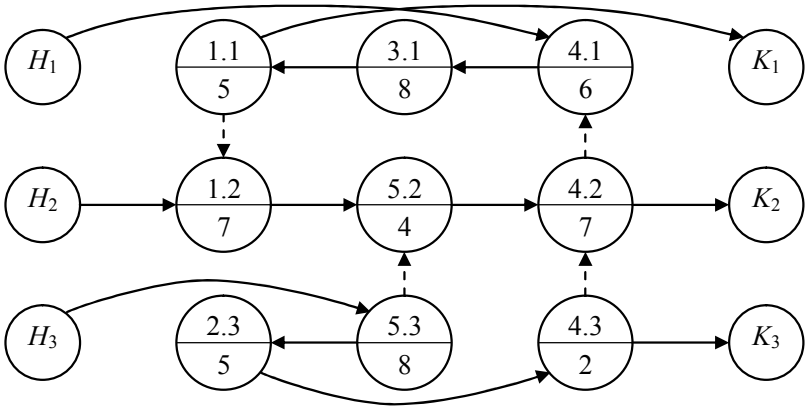

Рис. 3

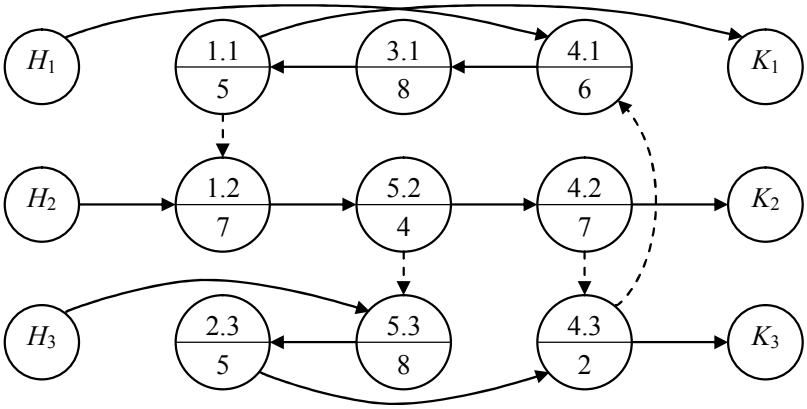

Рис. 4

Вариант 3. Соответствующая сеть приведена на рис. 5.

Установившиеся индексы приведены ниже:

Задача 1: $\lambda_{41} = 6$, $\lambda_{31} = 14$, $\lambda_{11} = 19$;

Задача 2: $\lambda_{12} = 7$, $\lambda_{52} = 12$, $\lambda_{42} = 19$;

Задача 3: $\lambda_{53} = 8$, $\lambda_{23} = 13$, $\lambda_{43} = 21$.

Имеем: $T_{\max} = 21 < 26$.

Вариант 3 имеет минимальное время среди всех вариантов. Этот вариант в данном случае является глобально-оптимальным.

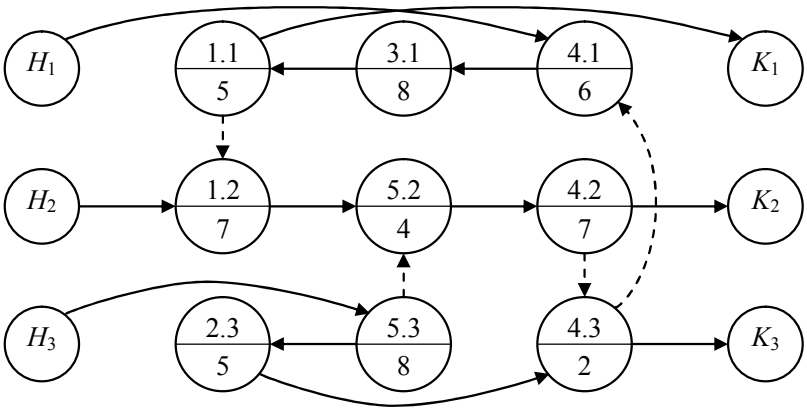

Рис. 5

Рассмотрим применение метода локальной оптимизации для задачи 2.

Определим

$$\delta_i = \min_k \Theta_k - \Theta_i$$

и припишем дугам, входящим в конечные вершины задач, длины δ_i. Покажем, что при таком преобразовании задача 2 сводится к задаче 1. Действительно, пусть $T_{\min}$ – максимальная продолжительность завершения всех задач в оптимальном решении преобразованной сети. Пусть $\Delta_{\min}$ – минимальное Δ, при котором существует решение такое, что для всех задач имеет место

$$T_i - \Theta_i \le \Delta_{\min},$$

где T_i – продолжительность решения i-й задачи. Имеем

$$T_i + \delta_i \le \Theta_i + \delta_i + \Delta_{\min} = \Theta_{\max} + \Delta_{\min},$$

откуда следует, что все задачи решаемы за минимальное время

$$T_{\min} = \Theta_{\max} + \Delta_{\min}.$$

Поэтому алгоритм локальной оптимизации, описанный выше, применим для решения задачи 2.

Пример 3. Пусть $\Theta_1 = 18$, $\Theta_2 = 17$, $\Theta_3 = 25$, $\delta_1 = 7$, $\delta_2 = 8$, $\delta_3 = 0$. Возьмем оптимальное решение (вариант 3 из примера 2). Установившиеся индексы будут те же самые, однако

$\Delta = \max(19–18, 19–17, 21–25) = 2$.

Сравнивая все четыре уже рассмотренных варианта легко видеть, что оптимальным является вариант 2, в котором

$T_1 = 19$, $T_2 = 18$, $T_3 = 26$,

$\Delta = \max(19–18, 18–17, 26–25) = 1$.

3. Учет «обходных путей»

Примем, что для каждой задачи имеется обходной путь ее решения, т. е. алгоритм, не связанный с проблемными вершинами вычислительной сети. Время решения задачи i по обходному пути обозначим R_i (очевидно, что R_i больше, чем время решения задачи i в вычислительной сети без учета ресурсных зависимостей, в противном случае задача сразу решалась бы по «обходному пути»).

Описание алгоритма.

Решаем задачу методом локальной оптимизации, как описано выше. Если

$$T_{\min} < R_{\min} = \min_i R_i ,$$

то задача решена. В противном случае удаляем все задачи, для которых $R_i = R_{\min}$ и повторяем алгоритм локальной оптимизации для оставшегося множества задач Q_1. Если

$$T_{\min}(Q_1) \le \min_{i \in Q_1} R_i ,$$

то задача решена. В противном случае удаляем все задачи, для которых

$$R_i = \min_{i \in Q} R_i ,$$

и повторяем алгоритм локальной оптимизации для оставшегося множества задач Q_2 и т. д.

За конечное число шагов s (не более m) будет получено решение такое, что

$$T_{\min}(Q_s) \le \min_{i \in Q_s} R_i .$$

Задача решена.

Пример 4. Пусть $R_3 = 18$, $R_2 = 19$, $R_1 = 20$. Так как $T_{\min}$ в примере 3 равно $21 > 18$, то удаляем задачу 3, решая ее по обходному пути. Имеем $Q1 = (1, 2)$. Легко показать, что при приоритетах $\pi_1 = (2, 1)$, $\pi_2 = (1, 2)$ получаем

$$T_{\min}(Q_1) = 19 = R_2 .$$

Задача решена. Минимальное время $T_{\min} = 19$, причем задача 3 решается по обходному пути.

4. Методы ветвлений и ветвей и границ

В отличие от метода локальной оптимизации, в методе ветвлений осуществляется последовательное построение расписаний, двигаясь от начальных вершин задач к конечным [2]. При попадании в проблемную вершину приоритетность соответствующих задач определяется по критерию минимума нижней оценки времени выполнения всех задач.

Предварительно определим минимальные времена T_{ij}, требуемые для завершения решения задачи j после решения подзадачи в вершине (i, j), без учета ресурсных зависимостей.

Описание алгоритма.

1. Строим расписание выполнения задач, начиная с начальных задач.

2. При возникновении конфликтной ситуации формируем оценочную задачу. Сетевая структура оценочной задачи приведена на рис. 6 (номер проблемной вершины опущен).

Структура приведена для трех задач, которые могут решаться в проблемной вершине. Вершины помечены двумя числами. Первое число – это номер задачи, а второе – тип вершины. Вер-

шины (i, 1) соответствуют проблемной вершине, вершины (i, 2) соответствуют временам T_j, вершины (i, 3) соответствуют решению подзадачи по обходному пути, вершины (i, 4) соответствуют раннему моменту начала решения подзадачи в проблемной вершине. Соответствующие времена l_{ij} указаны в нижних половинах вершин. Так, например, $l_{11} = 5$ означает, что подзадача 1 выполняется в проблемной вершине за 5 единиц времени, $l_{12} = 7$ означает, что после выполнения подзадачи 1 в проблемной вершине потребуется еще минимум 7 единиц времени для завершения задачи, $l_{13} = 26$ означает, что существует обходной путь решения задачи за 26 единиц времени, а $l_{14} = 4$ означает, что решение подзадачи 1 в проблемной вершине можно начать через 4 единицы времени. Задача состоит в определении расписания, минимизирующего время решения всех задач.

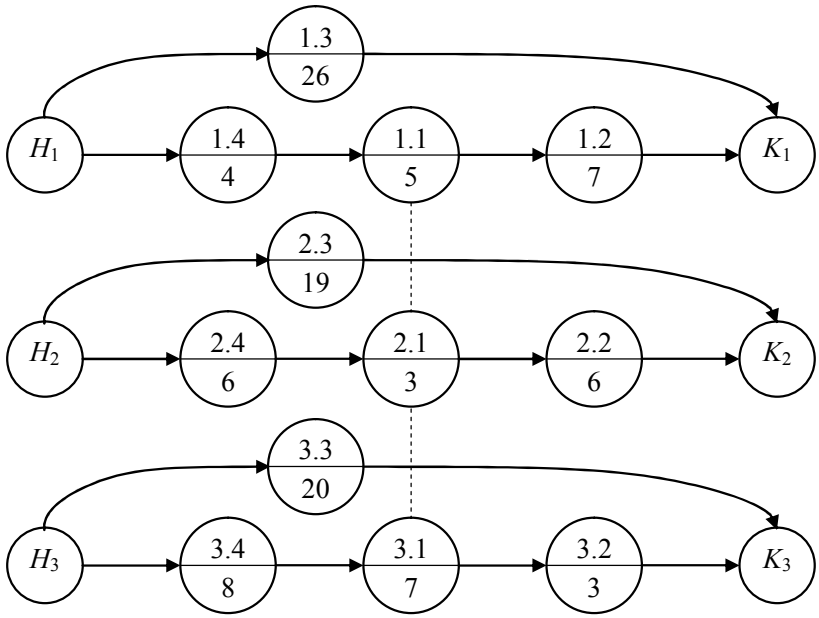

Рис. 6

Пусть число задач, решаемых в проблемной вершине, невелико, так что можно перебрать все возможные очередности выполнения подзадач (в примере рис. 1 их 6). Опишем алгоритм решения оценочной задачи при заданной очередности решения подзадач в проблемной вершине.

1 шаг. Определяем моменты окончания задач, применяя описанный выше модифицированный алгоритм определения кратчайших путей при заданных приоритетах задач. Если время решения всех задач

$$T_1 \le \min_k l_{k3},$$

то задача решена. В противном случае решаем задачи, для которых

$$l_{i3} \le \min_k l_{k3}$$

по обходному пути, исключая их из числа задач, решаемых в проблемной вершине.

2 шаг. Повторяем шаг 1 для оставшихся задач. Если время решения оставшихся задач

$$T_2 \le \min_{i \in Q_1} l_{i3},$$

где Q_1 – множество оставшихся задач, то задача решена. В противном случае исключаем задачи, для которых

$$l_{i3} \le \min_{i \in Q_1} l_{i3}$$

и повторяем шаг 1 для оставшихся задач и т. д.

Пример 5. Рассмотрим сеть, рис. 7.

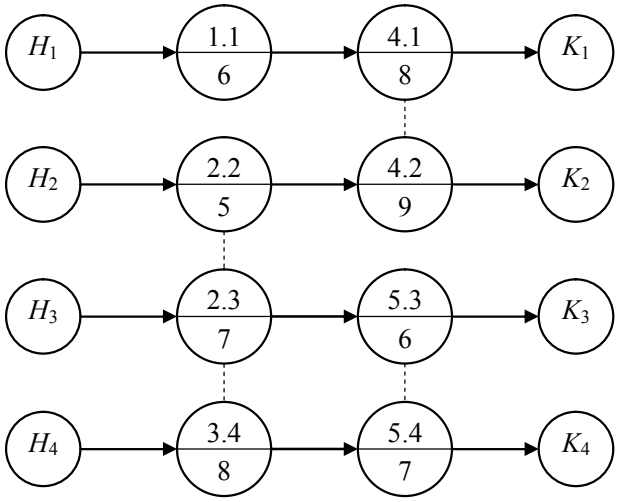

Рис. 7

В таблице указаны продолжительности решения задач по обходным путям.

j	1	2	3	4
l_j	18	21	15	22

Первый этап.

1. Момент времени $t_1 = 0$. Начинаем подзадачи (1.1) и (3.4). Конфликт возникает между задачами 2 и 3 (проблемная вершина 2). Поскольку $l_{24} = l_{34} = 0$, то в этом случае можно обойтись без перебора всех приоритетов. В этом случае первой следует начинать подзадачу с максимальным l_{j2}, т. е. в нашем случае подзадачу 2.

2. Момент времени $t_2 = 5$. Завершено решение подзадачи (2.2). Начинаем решение подзадачи (2.3).

3. Момент времени $t_3 = 6$. Завершено решение подзадачи (1.1). Возникает конфликт между задачами 1 и 2 (проблемная вершина 4). В этом случае также можно обойтись без перебора приоритетов. В этом случае начинается решение подзадачи, которую можно начать раньше (не позже) других. Это подзадача (4.2).

4. Момент времени $t_4 = 8$. Решена подзадача (3.4).

5. Момент времени $t_5 = 12$. Завершено решение подзадачи (2.3). Возникает конфликт между задачами 3 и 4 (проблемная вершина 5). Первой начинаем решение подзадачи (5.4), поскольку ее можно начать раньше, чем решение подзадачи (5.3).

6. Момент времени $t_6 = 14$. Завершено решение подзадачи (4.2). Начинаем решение подзадачи (4.1). Заметим, что подзадача (4.1) будет решена в момент

$t_{41} = 14 + 8 = 22 > T_2 = 21$,

поэтому задачу 2 решаем по обходному пути.

Второй этап.

1. Момент времени $t_1 = 0$. Начинаем решение подзадач (1.1), (2.3) и (3.4).

2. Момент времени $t_2 = 6$. Завершено решение подзадачи (1.1). Начинаем подзадачу (4.1).

3. Момент времени $t_3 = 7$. Завершена подзадача (2.3).

4. Момент времени $t_4 = 8$. Завершено решение подзадачи (3.4). Возникает конфликтная ситуация в вершине 5. Первой начинаем решение подзадачи (5.3) в момент времени $t = 7$.

5. Момент времени $t_5 = 13$. Завершено решение задачи 3, $T_3 = 13$. Начинаем подзадачу (5.4).

6. Момент времени $t_6 = 14$. Завершено решение задачи 1, $T_1 = 14$.

7. Момент времени $t_7 = 20$. Завершено решение задачи 4, $T_4 = 20$.

Поскольку все T_j, $j = 1, 3, 4$, меньше l_j, алгоритм закончен.

Теорема. Решение оценочных задач в каждой конфликтной ситуации дает оценку снизу времени завершения оставшихся задач.

Доказательство следует из того факта, что при решении оценочных подзадач не учитываются будущие ресурсные зависимости.

Теорема позволяет превратить метод ветвлений в метод ветвей и границ.

Поскольку алгоритм получения нижних оценок был описан выше, иллюстрацию метода ветвей и границ рассмотрим на примере. Ограничимся случаем, когда в каждой конфликтной ситуации участвует не более двух задач.

Пример 6. Рассмотрим сеть, рис. 8.

Рассмотрим сначала задачу без обходных путей.

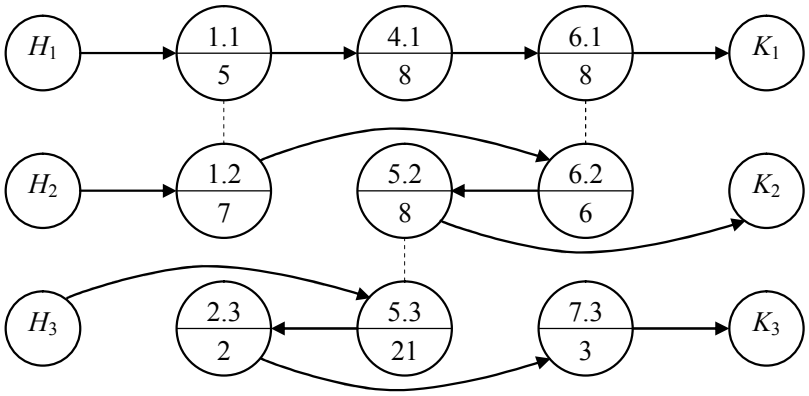

Рис. 8

Если имеются обходные пути, то решение можно улучшить. Так, например, если для задачи 2 имеется обходной путь с временем решения задачи 26, то конфликтные ситуации не возникают, и время решения всех задач составит 26 единиц. Можно поставить задачу разработки обходных путей решения задач при заданных зависимостях стоимости создания обходного пути для решения *j*-й задачи

Заключение

Рассмотренные задачи можно обобщить как по постановке, так и по методам решения. Так одна и та же задача может решаться разными способами. Этой ситуации соответствует уже не путь, а некоторая частичная подсеть вычислительной сети. Представляет интерес и исследование частных случаев, например, вычислительные сети с одной или двумя проблемными вершинами. Это и другие обобщения будут рассмотрены в последующих публикациях.

Литература

1. Buyya, R. Economy driven resource management architecture for computational power grids / R. Buyya, D. Abramson, J. Giddy // PDPTA '00: International Conference on Parallel and Distributed Processing Techniques and Applications, 2000.

2. Сигал, И.Х. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы: учеб. пособие / И.Х. Сигал, А.П. Иванова. – 2-е изд. испр. и доп. – М.: Физматлит, 2007. – 304 с.

Бурков Владимир Николаевич, д-р техн. наук, профессор, заведующий лабораторией 57, Институт проблем управления им. В. А. Трапезникова РАН; vlab17@bk.ru.

Ляпунцова Елена Вячеславовна, д-р техн. наук, профессор, помощник члена Совета Федерации РФ, Комитет Совета Федерации по социальной политике; председатель координационного совета, МРО «Лига преподавателей высшей школы»; lev77@me.com.

Шихалиев Руслан Сираджеддинович, аспирант, Московский государственный университет путей сообщения (МИИТ); shihaliev@live.ru.

Поступила в редакцию 29 февраля 2016 г.

DOI: 10.14529/ctcr160201

PROBLEM OF CALCULATIONS OPTIMIZATION IN THE NETWORK STRUCTURE

V.N. Burkov*[1]*, vlab17@bk.ru,
E.V. Lyapuntsova*[2, 3]*, lev77@me.com,
R.S. Shikhaliev*[4]*, shihaliev@live.ru

[1] V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russian Federation,
[2] Federation Council Committee on Social Policy, Moscow, Russian Federation,
[3] IPO "League of high school teachers", Moscow, Russian Federation,
[4] Moscow State University of Railway Engineering (MIIT), Moscow, Russian Federation

A computer network consisting of n vertices (vertices, which solved some problems), m input vertices and m output vertices (m is the number of tasks) is considered. Each task matches to way in the network with input H and output K, corresponding to some algorithm of problem solving. At the same time only one task can be solved in each vertex. Therefore, it may be a conflict in the moment of arrival to the vertex of a task if this vertex is busy with another task. The vertices in which several tasks may be solved at the same time, will be called problem vertices. The problems of tasks scheduling according to criteria of minimizing the time required to solve all the problems or minimize the maximum deviation from the required solution time are examined. Methods of local optimization, branch, branch and bound are proposed for their solution.

Keywords: network structures, problem vertices, problem of drawing up schedules, local optimization.

References

1. Buyya R., Abramson D., Giddy J. Economy Driven Resource Management Architecture for Computational Power Grids. Available at: http://www.buyya.com/papers/GridEconomy.pdf.

2. Sigal I.H., Ivanova A.P. *Vvedenie v prikladnoe diskretnoe programmirovanie: modeli i vychislitel'nye algoritmy* [Introduction to Applied Discrete Programming: Models and Computational Algorithms]. Moscow, Fizmatlit Publ., 2007. 304 p.

Received 29 February 2016

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Бурков, В.Н. Задача оптимизации вычислений в сетевых структурах / В.Н. Бурков, Е.В. Ляпунцова, Р.С. Шихалиев // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». – 2016. – Т. 16, № 2. – С. 5–14. DOI: 10.14529/ctcr160201

FOR CITATION

Burkov V.N., Lyapuntsova E.V., Shikhaliev R.S. Problem of Calculations Optimization in the Network Structure. *Bulletin of the South Ural State University. Ser. Computer Technologies, Automatic Control, Radio Electronics*, 2016, vol. 16, no. 2, pp. 5–14. (in Russ.) DOI: 10.14529/ctcr160201